Do It Yourself: Generic Pseudo-labeling for Domain Adaptation in Deep Computer Vision

Tsirmpas Dimitris

13th May 2024

1 Introduction

2 Related Works

3 Pseudo-labeling for Classification Enhancement

3.1 Model-agnostic Pseudo-labeling algorithm

We use a modified version of the iCAN algorithm [1]. The original iCAN algorithm was designed around a model created with the explicit purpose of learning domain-invariant features between the source and target datasets (CAN model). When the unmodified algorithm uses a generic deep model however a number of issues arise. Our approach seeks to remedy the most major of those issues.

First of all, in the original paper, the authors propose adding the loss of the fully supervised dataset $\mathcal{L}_{source}$, the loss of the target dataset $\mathcal{L}_{tar}$ and the loss of their model's domain-agnostic penalty $\mathcal{L}_{CAN}$ for each mini-batch as $\mathcal{L} = \mathcal{L}_{source} + \mathcal{L}_{tar} + \mathcal{L}_{CAN}$. This design decision most likely exists because of the need to have both a source and a target mini batch loaded on the network in order to calculate $\mathcal{L}_{CAN}$. Since this penalty does not exist in our algorithm, we instead split the training epoch into distinct backward passes for the source and target mini-batches, in order to reduce GPU VRAM requirements.

Secondly, the original iCAN algorithm selects pseudo-labeled batches for each backward pass because of the aforementioned mini-batch requirements. Since our algorithm proves much more unstable, as the underlying generic model may not learn domain-agnostic features, we instead perform pseudo-labeling once every N_{period} epochs. This mechanism ensures that our model will have acquired knowledge of the target distribution from the previously selected pseudo-labeled samples, before being asked to perform pseudo-labeling on the less-easily classifiable, unlabeled samples.

Thirdly, we do not use the ”domain bias reweighing function” used by the original authors when calculating $\mathcal{L}_{tar}$. Aside from necessitating a second ”domain” classifier, the sampling strategy we employ is inverse to the one proposed by the authors. iCAN attempts to select samples that do not fit the source distribution, in order to prevent its model from selecting target samples that are very similar to the source dataset (since they would score higher confidence scores). Our model-agnostic algorithm attempts to select samples that are closer to the source distribution and, as the model becomes more accustomed to the target distribution, slowly include samples closer to the latter. This is also the motivation behind not re-labeling pseudo-labeled samples.

The modified procedure can be found in Algorithm 1, where D_{source} is the source dataset containing the training instances, D_{target} is the equivalent target dataset and $\%$ is the modulo operator.

Algorithm 1 Modified general incremental learning algorithm

```
1: Train model on dataset $D_{source}$
2: $D_{pseudo}$ = $\{\}$
3: for epoch do
4:     if $epoch \% N_{period} = 0$ then
5:         Calculate $accuracy$ on the validation source dataset
6:         $\mathcal{T} = adaptive\_threshold(accuracy, \rho)$
7:         for each $d \in D_{target}$ do
8:             $label, confidence = model(d)$
9:             if $confidence > \mathcal{T}$ then
10:                $D_{pseudo} = D_{pseudo} \cup \{d : label\}$
11:                $D_{target} = D_{target} - \{d\}$
12:            end if
13:        end for
14:    end if
15:    $D_{rand\_source}$ = $\{\}$
16:    Select random samples from $D_{source}$ and add to $D_{rand\_source}$ such as $|D_{rand\_source}| = |D_{pseudo}|$
17:    Train epoch on $D_{rand\_source}$
18:    Train epoch on $D_{pseudo}$
19: end for
```

3.2 Adaptive threshold

We use the same `adaptive threshold` function used in the original paper, which adjusts the confidence threshold used to decide whether to pseudo-label a sample. The function is defined as $adaptive\_threshold(acc, \rho) = \frac{1}{1+e^{-\rho * acc}}$, where acc is the accuracy of the classifier, and ρ a tunable hyperparameter, with $\rho = 3$ in the original paper. We evaluate the classifier accuracy on the validation set of the source dataset.

Higher ρ values lead to a steeper decision curve, as see in Figure. The ρ parameter is thus a very convenient way of tuning the pseudo-labeling procedure; a high ρ value leads to samples chosen conservatively, ensuring that more samples are correctly labeled, while lower values lead to more samples overall being chosen, but with more incorrect labels. Since our strategy does not involve re-labeling of pseudo-labeled samples, and because of the lack of external mechanisms to detect outliers, we recommend that $\rho \in [3, 4]$, although the value is dependent on the datasets and underlying classifier.

Overall, the adaptive threshold function provides us with an easy-to-understand mechanism which both adapts to the current state of the model, and which can be tuned with a single hyperparameter. This hyperparameter however remains of crucial importance, and should be tuned either by the number of accepted samples, or if available, a target validation dataset which tracks the pseudo-labeling misclassifications.

References

[1] Weichen Zhang et al. 'Collaborative and Adversarial Network for Unsupervised Domain Adaptation'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2018, pp. 3801–3809. DOI: 10.1109/CVPR.2018.00400.